Notasi asimtotik

Bahasan

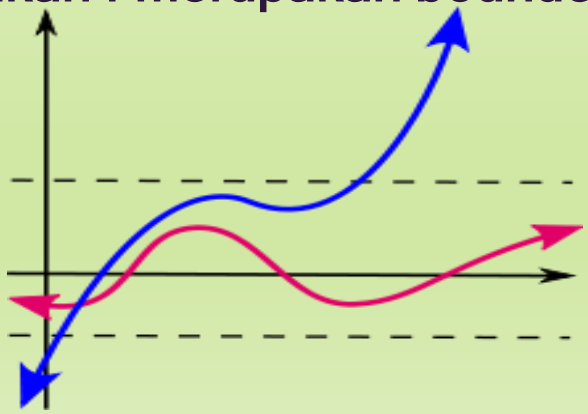
- **Notasi asimtotik**
  - ***O* (big oh)**
  - **Ω (big omega)**
  - **Θ (big theta)**

Notasi Asimtotik

- **Misalkan pengurutan suatu record panjang n item dengan suatu algoritma waktu f(n).**
- **Berapa lama waktu yang dibutuhkan untuk mengurutkan n record?**
- **Untuk n semakin besar?**
  - **apakah f tetap?**
  - **Akankah f tumbuh secara linier?**
  - **Akankah f tumbuh secara eksponensial?**
- **Untuk itu perlu diketahui seberapa cepat f tumbuh terhadap n.**

Notasi asimtotik

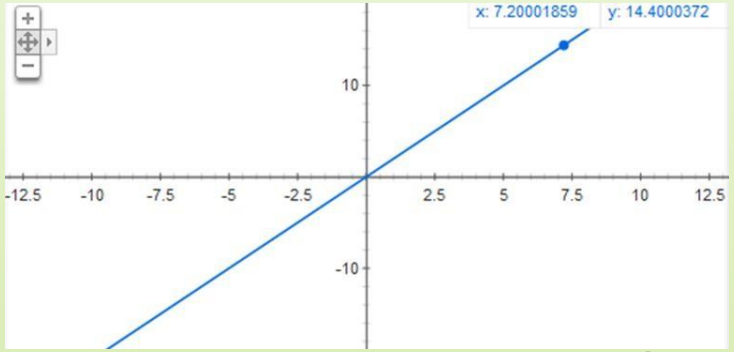
- Notasi asimtotik merupakan himpunan fungsi yang dibatasi oleh
  - suatu fungsi, dengan n yang cukup besar.
- Misal terdapat n elemen array, dan kita ingin sorting dengan suatu algoritma yang memiliki kompleksitas f(n). Berapa lama waktu yang dibutuhkan untuk sorting dengan input size (n) tersebut?
- Bagaimana jika nilai n semakin besar?
- Kita akan berfokus pada interval nilai n dari fungsi kompleksitas suatu algoritma ketika nilai n semakin besar (atau masuk pada nilai n yang besar).
- Apakah f merupakan bounded function?



- bounded function (merah)
- unbounded function (biru)

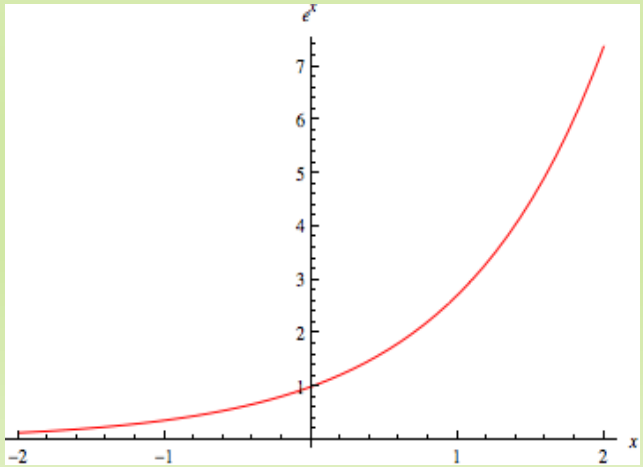
Notasi asimtotik

- **Apakah f tumbuh secara linear?**



- Misal, f(x)=2x
- Fungsi linier adalah fungsi polynomial dengan derajat 0 atau 1

- **Apakah f tumbuh secara eksponensial?**



- Tujuan Asymptotic Notation adalah untuk mengetahui seberapa cepat pertumbuhan / order of grow (OoG) dari fungsi f, sehubungan dengan semakin naiknya nilai n.

Notasi asimtotik

Memperkirakan formula untuk run-time

Indikasi kinerja algoritma
(untuk jumlah data yang sangat besar)

- **Misalkan:**

  $T(n) = 5n^2 + 6n + 25$

  T(n) proporsional untuk ordo n^2 untuk data yang sangat besar.

Notasi Asimtotik

- Indikator efisiensi algoritma berdasar pada OoG pada basic operation suatu algoritma.
- Penggunaan notasi sebagai pembanding urutan OoG:
  - O (big O)
  - Ω (big omega)
  - Θ (big theta)
- t(n) : fungsi running time suatu algoritma (diindikasikan dengan *basic operation count* (C(n))
- g(n) : fungsi running time lain sebagai pembanding dengan t(n)

Klasifikasi fungsi berdasarkan ting pertumbuhan asimtotiknya

- Mengklasifikasikan fungsi *running time* dari tingkat pertumbuhan Asimtotiknya.
- Tingkat pertumbuhan asimtotik / tingkatan (order) asimtotik
  - / tingkatan fungsi running time dari suatu algoritma.
    - Membandingkan dan mengklasifikasikan fungsi dengan mengabaikan faktor konstan (c) dan *input size* (n) yang kecil.
- Beberapa macam kelas Notasi Asimtotik :
  - **O(g(n)), Asimtotik batas Atas**;
  - **Ω(g(n)), Asimtotik batas bawah**
  - **Θ(g(n)), Asimtotik batas atas dan batas bawah**

Contoh

Contoh: $f(n) = n^2 - 5n + 13$.

- Konstanta 13 tidak akan pernah berubah sebagaimana n, sehingga dianggap tidak berpengaruh secara signifikan ketika nilai n semakin meningkat dan bernilai sangat besar. Kemudian orde terendah -5n pun tidak akan banyak berpengaruh terhadap f, jika dibandingkan dengan n^2.

- Kita akan menunjukkan bahwa $f(n) = \Theta(n^2)$.
- Q: Apakah arti dari $f(n) = \Theta(g(n))$?
- A: Secara intuitif (tanpa analisis secara matematis), itu berarti bahwa fungsi f memiliki tingkatan pertumbuhan yang sama besarnya dengan fungsi g.

Contoh (lanj.)

- Q: Apakah arti dari $f_1(n) = \Theta(1)$?
- A: $f_1(n) = \Theta(1)$ berarti bahwa setelah dijalankan beberapa nilai n, f_1 ini batas atas & bawahnya dibatasi dengan sebuah konstanta yang bernilai 1.
- Q: Apakah arti dari $f_2(n) = \Theta(n \log n)$?
- A: $f_2(n) = \Theta(n \log n)$ berarti bahwa setelah dijalankan beberapa nilai n, f_2 ini memiliki tingkatan pertumbuhan yang sama besarnya dengan n log n.
- Secara umum, $f(n) = \Theta(g(n))$ berarti bahwa f(n) adalah member dari $\Theta(g(n))$ dimana $\Theta(g(n))$ merupakan sekumpulan fungsi yang memiliki tingkatan pertumbuhan running time yang sama besarnya(setara).

O(g(n)) Big-Ooh (≤)

- **Simbol O diperkenalkan pada tahun 1927 untuk menunjukkan pertumbuhan relatif dari dua fungsi berdasarkan perilaku asimtotik dari fungsi.**
- **O(g(n)) adalah himpunan semua fungsi dengan orde pertumbuhan yang lebih kecil atau sama dengan g(n)**
- **Contoh :**
  - $n \in O(n^2)$; $100n + 5 \in O(n^2)$
  - $\frac{1}{2} n (n-1) \in O(n^2)$
  - $n^3 \notin O(n^2)$; $0.0001\, n^3 \notin O(n^2)$; $n^4+n+1 \notin O(n^2)$

Upper Bound Notation (Big-O)

- Fungsi Umum
  - f(n) adalah O(g(n)) jika ∃ konstanta positif ***c*** dan n_0
  - sedemikian hingga $f(n) \leq c \cdot g(n)$ ∀ $n \geq n_0$
- Misal jika $f(n)=1000n$ dan $g(n)=n^2$, $c=1$, $n_0=1000$, $n \geq 1000$, maka $f(n_0) \leq 1.g(n_0)$ →(True). Sehingga kita dapat mengatakan bahwa f(n) = O(g(n))
- Cara lainnya :

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{1000n}{n^2} = \lim_{n\to\infty} \frac{1000}{2n} = \lim_{n\to\infty} \frac{500}{n} = \frac{500}{\infty} = 0$$

Maka, $1000n \in O(n^2)$

Big-O, the Asymptotic Upper Bound

- Big-O merupakan notasi yang paling populer untuk ukuran run time, karena kita biasanya pasti mencari **worst case time.**
- Jika Running Time dari Algorithm X adalah $O(n^2)$, maka untuk berapapun input size-nya (n), running time dari algoritma X adalah fungsi kuadrat, untuk n yang cukup besar.

- Misal $2n^2 = O(n^3)$.
- Dari definisi sebelumnya, misal dengan $c = 1$ dan $n_0 = 2$. Tetapi jika $O(n^2)$ digunakan, maka akan lebih presisi / dekat daripada $O(n^3)$.

Klasifikasi Fungsi : BIG-O (1/2)

- Sebuah fungsi f(n) dikatakan sebagian besar tingkat pertumbuhannya paling mendekati dengan fungsi logaritmik, jika $f(n) = O(\log n)$
- Sebuah fungsi f(n) dikatakan sebagian besar tingkat pertumbuhannya paling mendekati dengan fungsi kuadratik, jika $f(n) = O(n^2)$
- Sebuah fungsi f(n) dikatakan sebagian besar tingkat pertumbuhannya paling mendekati dengan fungsi polynomial, jika $f(n) = O(n^k)$, untuk bilangan asli $k > 1$
- Sebuah fungsi f(n) dikatakan Sebagian bersar tingkat pertumbuhannya paling mendekati dengan fungsi exponential, jika terdapat konstanta c, sedemikian hingga $f(n) = O(c^n)$, dan $c > 1$
- Sebuah fungsi f(n) dikatakan tingkat pertumbuhannya mendekati fungsi faktorial, jika $f(n) = O(n!)$.

Klasifikasi Fungsi : BIG-O (2/2)

- **Sebuah fungsi f(n) dikatakan memiliki running time konstran, jika ukuran input n tidak berpengaruh pada running time dari algoritma (misalnya, algoritma yang hanya memiliki proses untuk memasukkan nilai ke suatu variabel). Maka persamaan kompleksitas algoritma tersebut adalah f(n) = c**

- **Bentuk klasifikasi fungsi logaritmik lainnya :**
  - f(n) = O(n log n)
  - f(n) = O(log log n)

O-notation: Formally

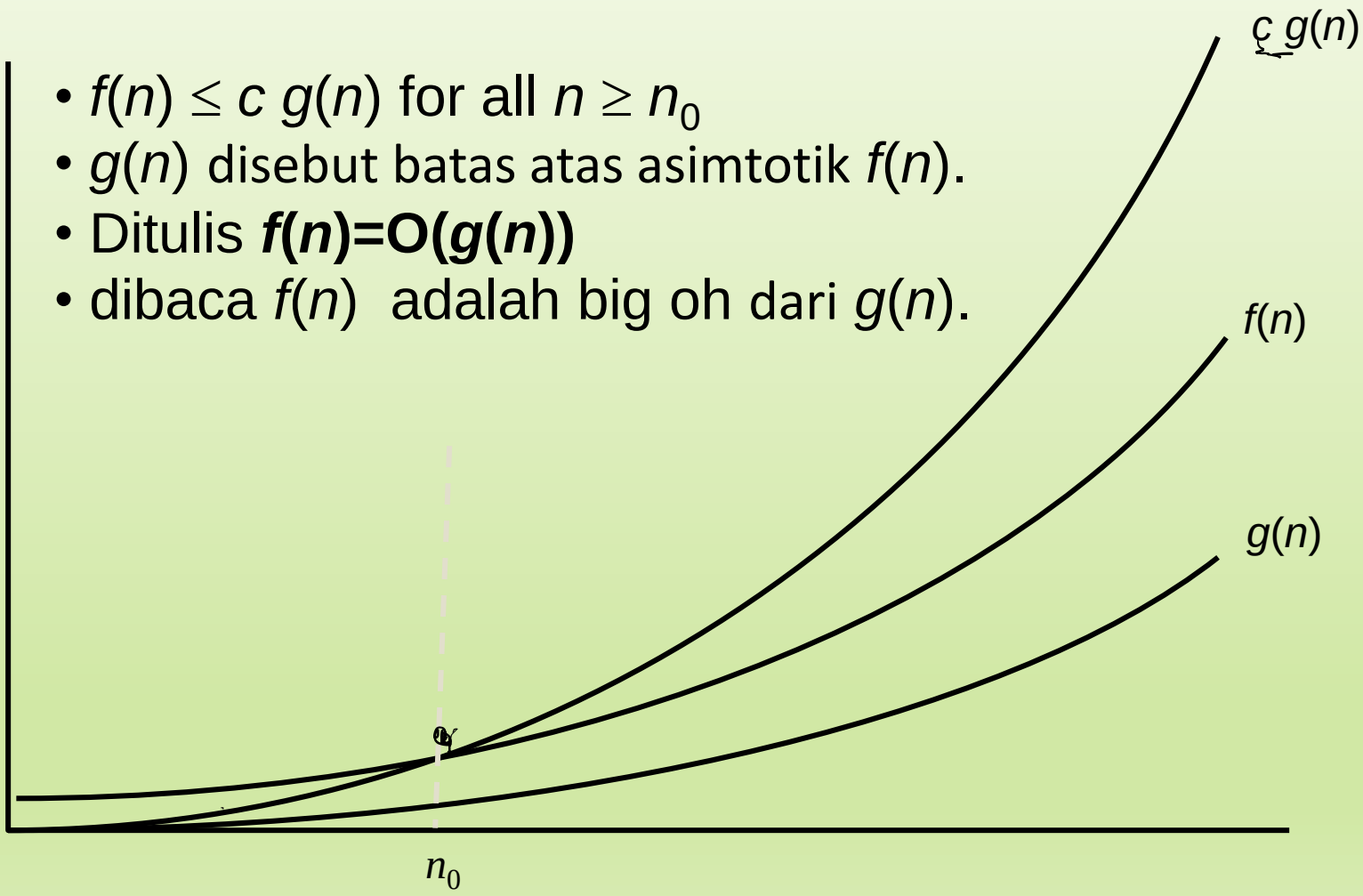
- **DEF1: suatu fungsi t(n) disebut dalam *O*(g(n)), t(n) $\in$ *O*(g(n)), jika t(n) dibatasi atas oleh beberapa konstanta kali g(n) untu semua n besar**

- **i.e. terdapat beberapa konstanta positif c dan beberapa bilangan bulat positif n_0, sehingga**

$$t(n) \leq cg(n) \text{ untuk semua } n \geq n_0$$

O(g(n)) (Batas atas ≤)

- $f(n) \leq c\ g(n)$ for all $n \geq n_0$
- $g(n)$ disebut batas atas asimtotik $f(n)$.
- Ditulis $\boldsymbol{f(n)=O(g(n))}$
- dibaca $f(n)$ adalah big oh dari $g(n)$.



Contoh O(g(n))

1. Tunjukkan bahwa : $100n + 5 \in O(n^2)$

- **carilah c dan n_0, sehingga $t(n) \leq cg(n)\ \forall\ n \geq n_0$**

  $100n + 5 \leq 100n + n\ (\forall\ n \geq 5)$

  $= 101n \leq 101n^2 \rightarrow c=101, n_0=5$

  $100n + 5 \leq 100n + 5n\ (\forall\ n \geq 1)$

  $= 105n \leq 105n2 \rightarrow c=105, n_0=1$

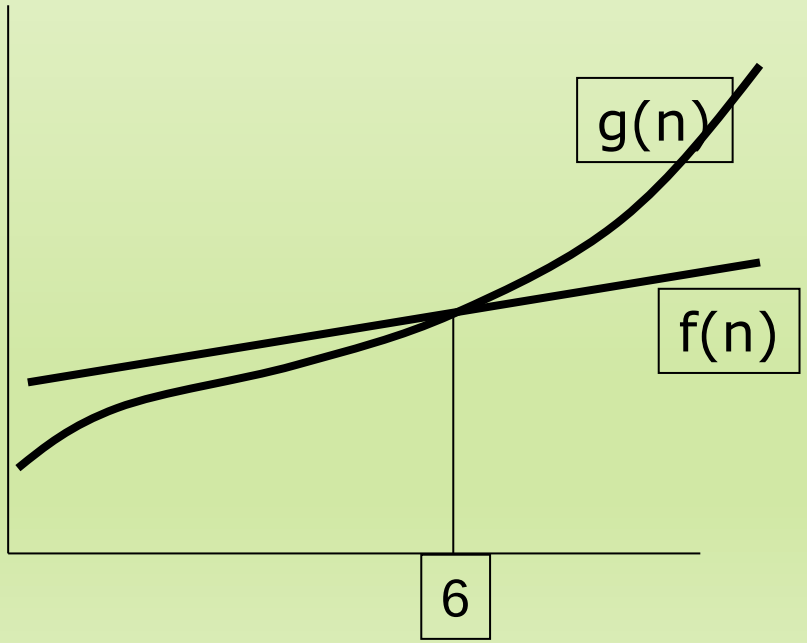
Contoh O(g(n))

2. **2n + 10 adalah O(n)**
   - $2n + 10 \leq cn$
   - $(c-2)n \geq 10$
   - $n \geq 10/(c-2)$
   - $c = 3$ dan $n_0 = 10$

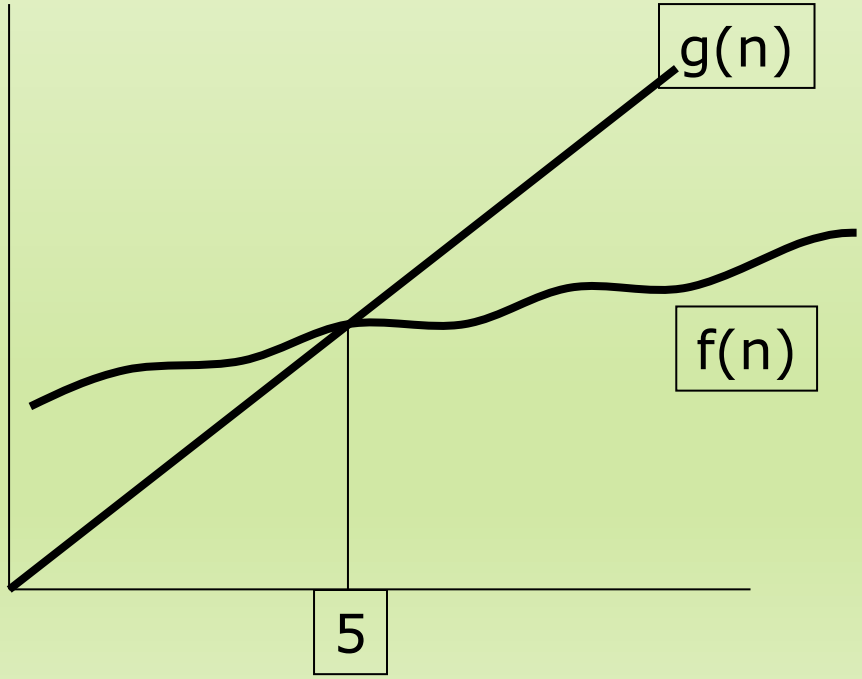
3. **7n – 2 adalah O(n)**
   - $C > 0$ dan $n_0 \geq 1$ sds $7n - 2 \leq c.n$ untuk $n \geq n_0$
   - $C = 7$ dan $n_0 = 1$

Contoh O(g(n))

4. $\forall n > 6$, $g(n) > 1\ f(n)$.
maka f adalah O(g(n)).
yakni $f(n) = O(g(n))$.



5. $\exists\ n_0=5$ sehingga. $\forall$ $n>n_0$, $f(n) < 1\ g(n)$.
Jadi, f(n) = O(g(n)).



Contoh O(g(n))

6, $g(n) = 4n^2$

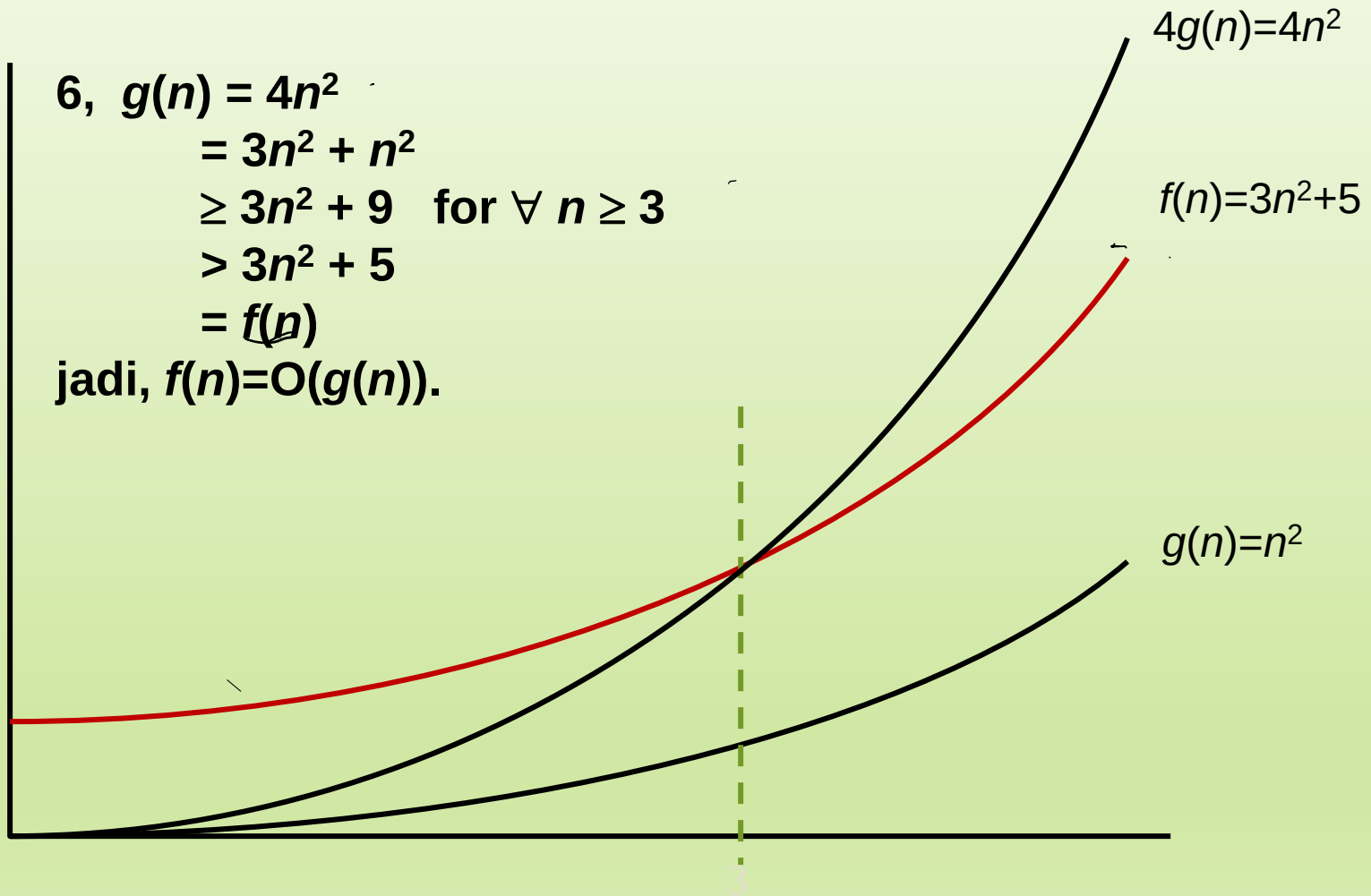
$= 3n^2 + n^2$

$\geq 3n^2 + 9$ for $\forall\ n \geq 3$

$> 3n^2 + 5$

$= f(n)$

jadi, $f(n)=O(g(n))$.



Latihan 1

Tunjukkan bahwa

1. $3n^2+2n+5 = O(n^2)$
2. $3n^3 + 20n^2 + 5 = O(n^3)$
3. $3 \log n + \log \log b = (O(\log n))$
4. $2n^3 + 3n^2 + n = O(n^4)$

Big Ω, Big-Omega (≥)

- **f(n) adalah Ω(g(n)) if ∃ *c > 0* dan n_0 sehingga**

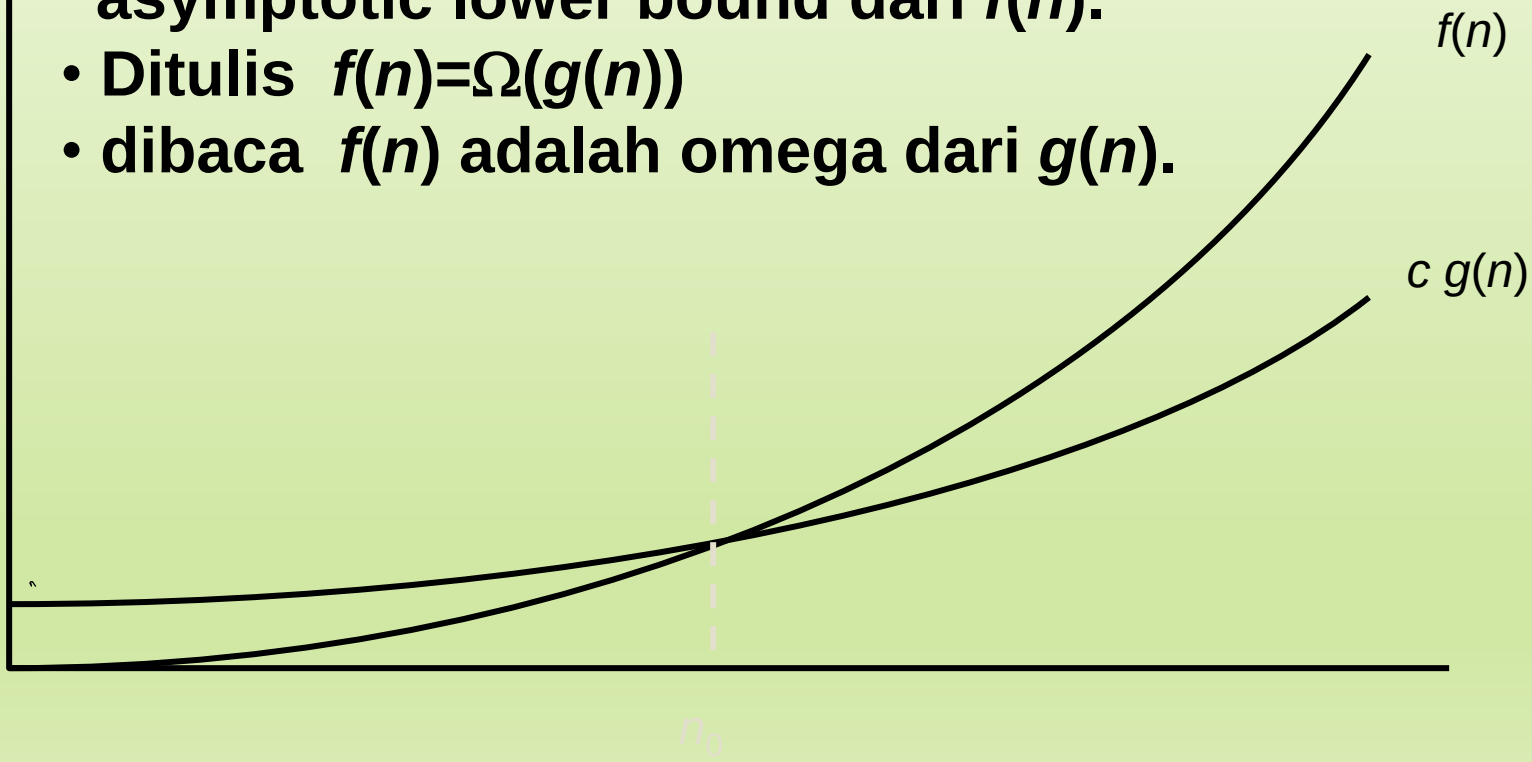
  $0 \leq c \cdot g(n) \leq f(n)$ ∀ $n \geq n_0$

- **Proof:**
  - **Misalkan run timenya adalah an + b**
  - **diasumsikan a dan b positive an ≤ an + b**

3 log n + log log b (O(log n))

Big Ω - Batas bawah

- $f(n) \geq c\, g(n)\ \forall\ n \geq n_0$
- $g(n)$ disebut asymptotic lower bound dari $f(n)$.
- Ditulis $f(n)=\Omega(g(n))$
- dibaca $f(n)$ adalah omega dari $g(n)$.



Ω-notation: Formally

- **DEF2: suatu fungsi t(n) disebut ada di Ω(g(n)), t(n) ∈ Ω(g(n)), if t(n) dibatasi bawah oleh beberapa konstanta kali g(n) untuk semua n besar.**

- **i.e. Terdapat beberapa c > 0 dan beberapa nonnegative integer n_0, sehingga**

$$t(n) \geq cg(n) \ \forall \ n \geq n_0$$

Contoh: Big Omega

2. $n^{1/2} = \Omega(\log n)$.

berdarakan definisi c = 1 dan $n_0 = 16$.

Misalkan $n \geq 16 : n^{1/2} \geq (1) \log n$

Jikka $n = (\log n)^2$

3. $0.5n^2 - 5n + 2 = \Omega(n^2)$.

Let c = 0.25 and $n_0 = 25$.

$0.5 n^2 - 5n + 2 = 0.25(n^2)$ for all n = 25

If $b = a^c <=> c = \log_a b$

a, b, c are *real numbers* and $b > 0, a > 0, a \neq 1$

a is called **"base"** of the logarithm.

Contoh Big tOmega

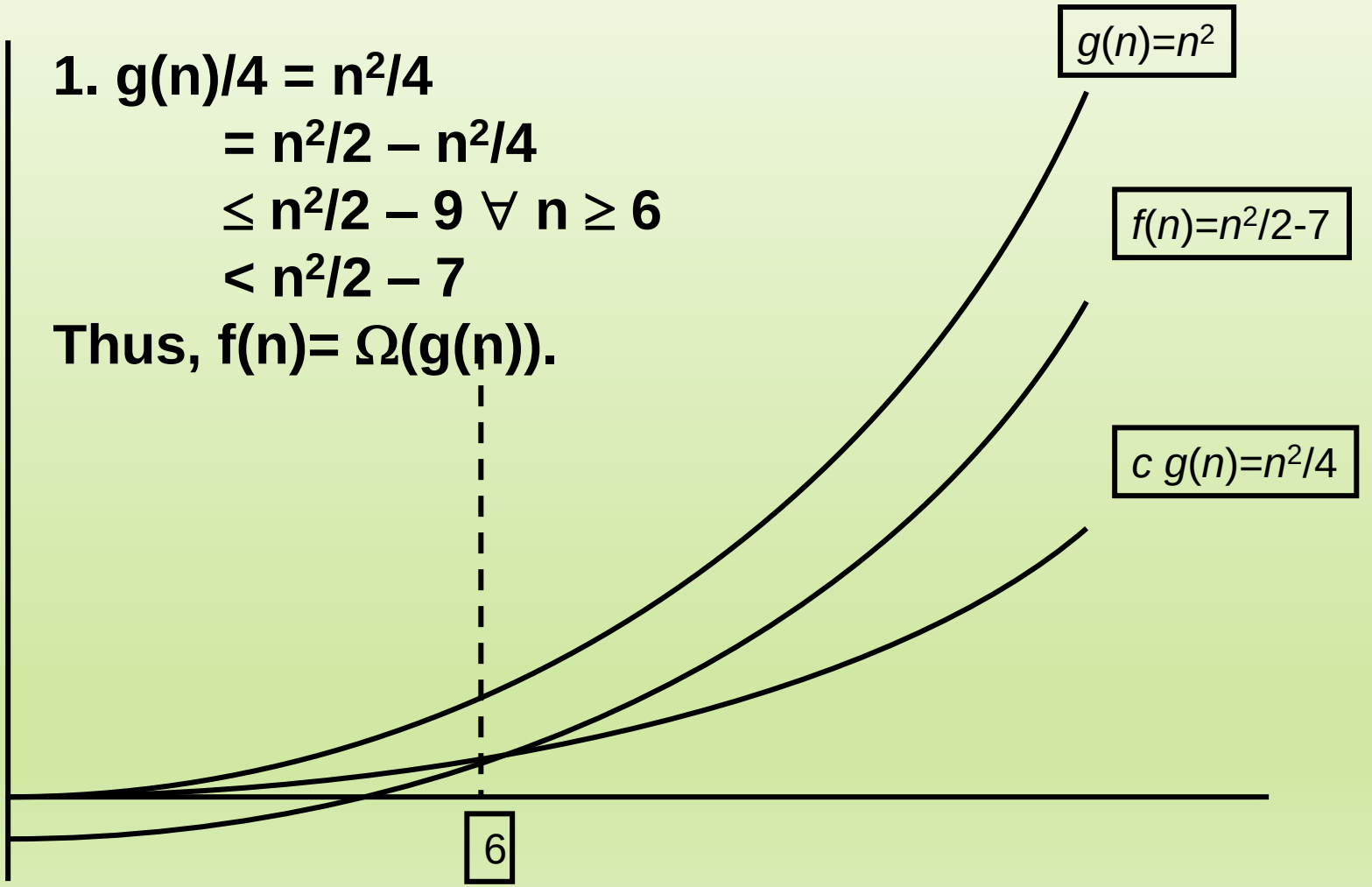
1. $g(n)/4 = n^2/4$

$= n^2/2 – n^2/4$

$\leq n^2/2 – 9 \; \forall \; n \geq 6$

$< n^2/2 – 7$

Thus, $f(n) = \Omega(g(n))$.



θ Big Theta

- **Dua fungsi f dan g dikatakan memiliki pertumbuhan yang sama, f = *Big Theta*(g) jikka keduanya**

  f= θ(g) dan g = θ(f).

- **Definisi f(n) = θ (g(n)) artinya $\exists\ c_1, c_2 > 0$ and n_0 sehingga**

  $$c_1\, g(n) \leq f(n) \leq c_2\, g(n)\ \forall\ n \geq n_0$$

- **jika f(n) = O(g(n)) dan f(n) = Ω(g(n)) then f(n)= θ(g(n))**

  (contoh. $f(n) = n^2$ dan $g(n) = 2n^2$)

θ-notation: Formally

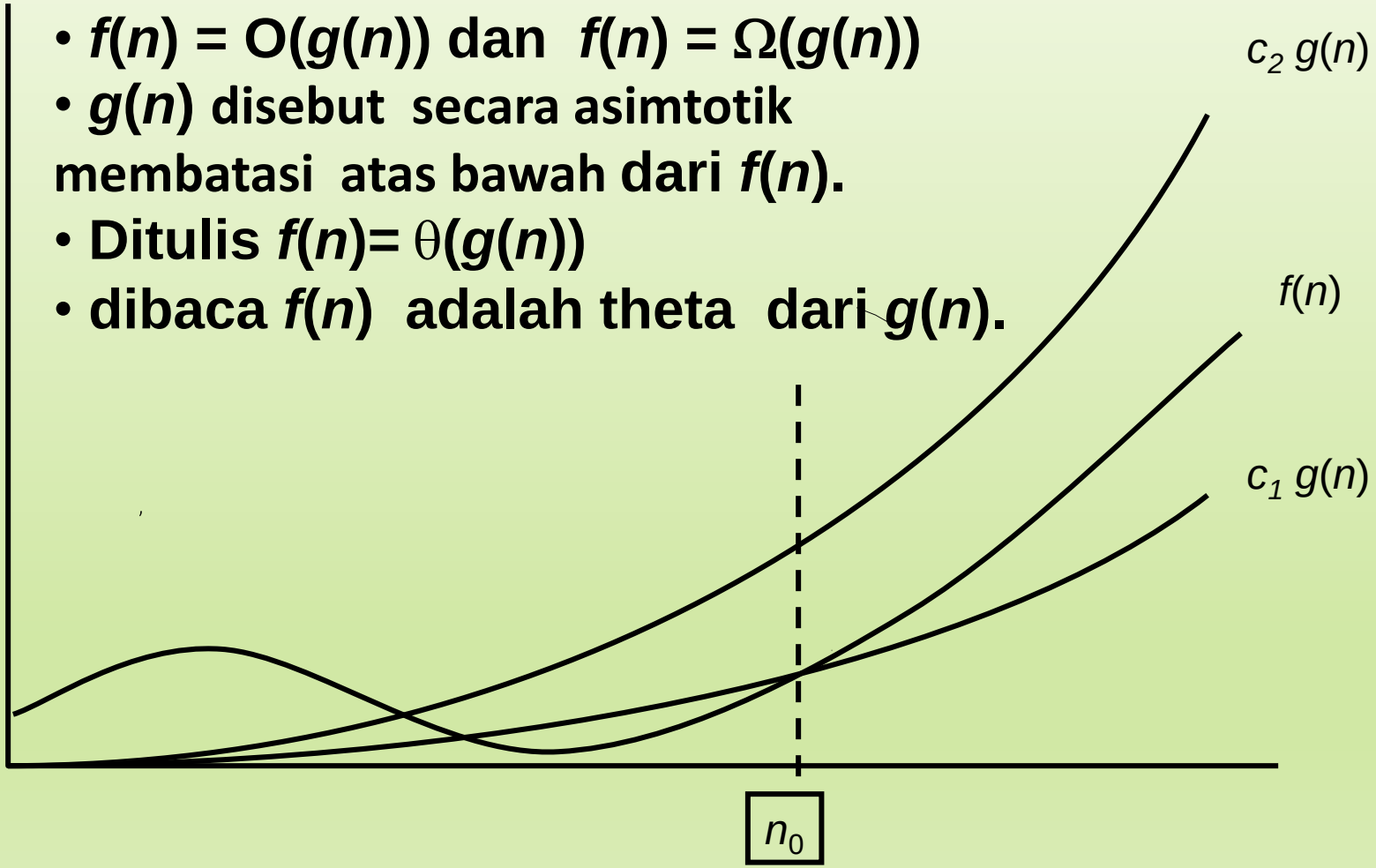
- **DEF3: Suatu fungsi t(n) dikatakan berada di $\theta(g(n))$, ditandai $t(n) \in \theta(g(n))$, jika t(n) dibatasi baik di atas maupun di bawah oleh beberapa kelipatan konstan g(n) untuk semua n besar**

- **i.e terdapat beberapa konstanta positif c_1 dan c_2 dan beberapa bilangan bulat nonnegatif n_0, sehingga**

$$c_2 g(n) \leq t(n) \leq c_1 g(n) \; \forall \; n \geq n_0$$

Big Theta

- $f(n) = O(g(n))$ dan $f(n) = \Omega(g(n))$
- $g(n)$ disebut secara asimtotik membatasi atas bawah dari $f(n)$.
- Ditulis $f(n) = \theta(g(n))$
- dibaca $f(n)$ adalah theta dari $g(n)$.



Contoh

1. **Buktikan baha: $\frac{1}{2}n(n-1) \in \theta(n^2)$**

- **Dari DEF3: cari c_1 dan c_2 dan beberapa nonnegative integer n_0, sehingga $c_2 g(n) \leq t(n) \leq c_1 g(n)\ \forall n \geq n_0$**

- **Batas atas: $\frac{1}{2}\, n(n-1) = \frac{1}{2}\, n^2 - \frac{1}{2}\, n \leq \frac{1}{2}\, n^2\ (\forall n \geq 0)$**
- **Batas bawah: $\frac{1}{2}\, n(n-1)$**

  $= \frac{1}{2}\, n^2 - \frac{1}{2}\, n \geq \frac{1}{2}\, n^2 - \frac{1}{2}\, n\ \frac{1}{2}\, n\ (f\forall\ n \geq 2) = \frac{1}{4}\, n^2$
- **$c_1 = \frac{1}{2}$, $c_2 = \frac{1}{4}$, $n_0 = 2$**

Examples

2. $2n^3 + 3n^2 + n = 2n^3 + 3n^2 + O(n)$

$= 2n^3 + \theta(n^2 + n)$

$= 2n^3 + \theta(n^2) = \theta\ n^3)$

3. $0.5\ n^2 - 5n + 2 = \theta(\ n^2)$

$n_0 = 25,\ c_1 = 0.25,\ c_2 = 0.5$

.

Other Asymptotic Notations

- **A function f(n) is o(g(n)) if ∃ positive constants *c* and n_0 such that**
  $f(n) < c\ g(n)\ \forall\ n \geq n_0$
- **A function f(n) is ω(g(n)) if ∃ positive constants *c* and n_0 such that**
  $c\ g(n) < f(n)\ \forall\ n \geq n_0$
- **Intuitively,**
  - o() is like <
  - ω() is like >
  - Θ() is like =
  - O() is like ≤
  - Ω() is like ≥

Interpretasi

- **Contoh: $f(n) = n^2 - 5n + 13$.**
  - **Konstanta 13 tidak berubah seiring bertambahnya n, jadi itu tidak penting. Suku orde rendah, -5n, tidak terlalu berpengaruh pada f dibandingkan suku kuadrat, n^2.**

Akan ditunjukkan bahwa $f(n) = \theta(n^2)$.

- **Q: Apa artinya mengatakan $f(n) = \theta(g(n))$?**
- **A: Secara intuitif, berarti bahwa fungsi f urutan pertumbuhannya sama dengan g.**

Interpretasi

Q: Apa artinya mengatakan $f_1(n) = \theta(1)$?

A: $f_1(n) = \theta(1)$ berarti setelah beberapa n, f1 dibatasi di atas & di bawah oleh konstanta.

Q: Apa artinya mengatakan $f_2(n) = \theta(n \log n)$?

A: $f_2(n) = \theta(n \log n)$ berarti setelah beberapa n, f_2 dibatasi di atas dan di bawah oleh waktu konstan n log n. Dengan kata lain, f_2 adalah orde besarnya yang sama dengan n log n.

- **Secara umum, $f(n) = \theta(g(n))$ artinya bahwa f(n) adalah anggota dari $\theta(g(n))$ dimana $\theta(g(n))$ adalah himpunan fungsi dari order sama.**

Latihan 2

1. **True or false:**
   a. $n(n+1)/2 \in O(n^3)$
   b. $n(n+1)/2 \in O(n^2)$
   c. $n(n+1)/2 \in \theta(n^3)$
   d. $n(n+1)/2 \in \Omega(n)$
   e. $0.25n^2 - 5n + 2 = \Omega(n^2)$.
   f. $0.25n^2 - 5n + 2 = O(n^2)$.
   g. $0.25n^2 - 5n + 2 = \theta(n^2)$

2. **Indicate the class $\theta(g(n))$:**
   a. $(n^2+1)^{10}$
   b. $(10n^2+7n+3)^{1/2}$
   c. $2n \log (n+2)^2+(n+2)^2 \log (n/2)$

Latihan 3

3. **Tentukan OoG dari masing-masing soal**
   a. **$f1(n) = 10\ n + 25\ n^2$**
   b. **$f2(n) = 20\ n \log n + 5\ n$**
   c. **$f3(n) = 12\ n \log n + 0.05\ n^2$**
   d. **$f4(n) = n^{1/2} + 3\ n \log n$**

Terimakasih...